

End-to-End (E2E) Testing

System testing

Acceptance testing

Acceptance Testing

- **Testing the entire system as a whole.**
 - **UI + Server-side + Database**
- **Concerns:**
 - **Functionality ******
 - **From the USER interface perspective.**
 - **Performance**
 - **Load/Stress**

E2E Testing

- **Many similarities to API testing:**
 - **Blackbox – not looking at internals, only expected output for specific inputs.**
 - **May also be interested in side-effects, e.g. database changes.**
 - **The Asynchronous nature (for web/mobile apps).**
- **Unit and Integration test should have ironed out (most) 'low level' errors.**

E2E Testing

- **Web apps - Targeting the browser interface.**
 - **Form submits.**
 - **Navigation.**
 - **Flows e.g. shopping cart checkout.**

Automation Tools

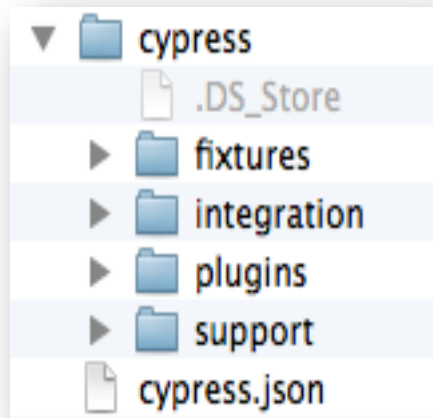
- **Traditional tool suite: Mocha + Chai + Selenium.**
- **(Very) modern tool suite: Cypress**
 - **Uses Mocha and Chai internally.**
- **Cypress.**
 - **Win / Mac / Linux**
 - **MIT License**
 - **Open Source**

Cypress

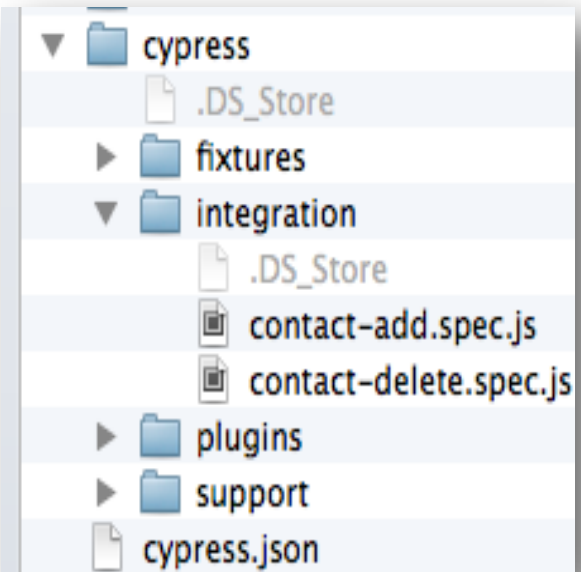
- **Getting started:**

`$ npm install --save-dev cypress`

- **(Default) Test code folder structure:**



- **fixtures** – mock data
- **Integration** – test code, termed specs.
- **cypress.json** – config file



- **CLI (Command Line Interface) has 2 main commands:**

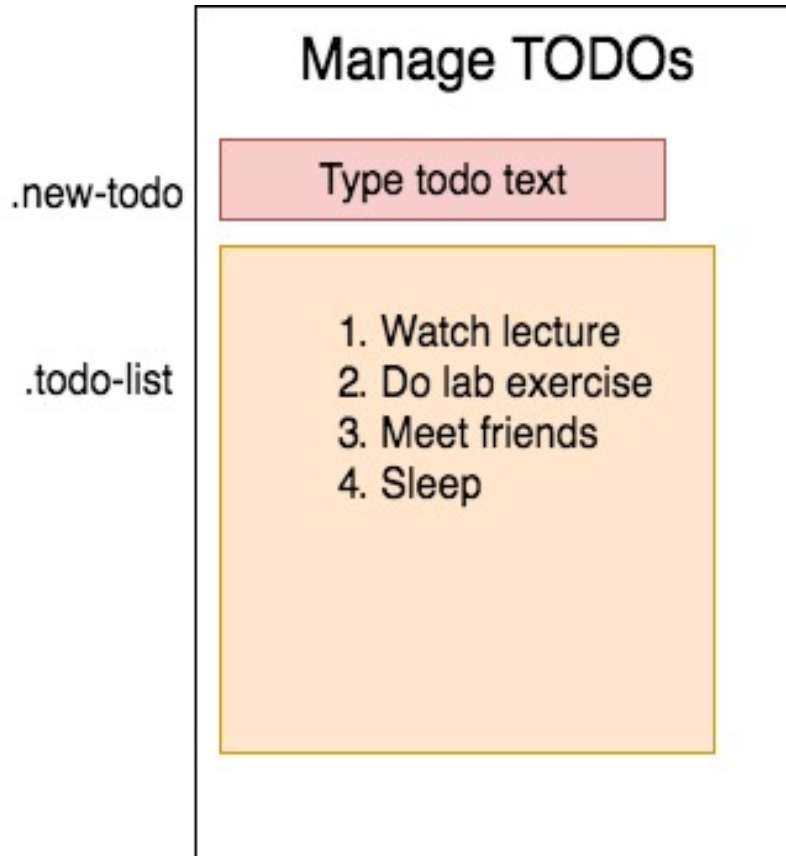
`$ npx cypress open`

- **GUI interactive mode (Dev)**

`$ npx cypress run`

- **headless mode (Testing/CI).**

Sample test code



```
describe("TODO app", () => {  
  it('adds 2 todos', () => {  
    cy.visit('http://localhost:3000')  
    cy.get('.new-todo')  
      .type('learn testing{enter}')  
      .type('complete lab{enter}')  
    cy.get('.todo-list li')  
      .should('have.length', 2)  
  })  
})
```

- **Method Chaining style e.g.**
cy(...).get(...).type(...)

Cypress statements

`cy.get(...selector ...)` `.should(...assertion/expectation)`

Command

Assertion (Expectations)

- **Commands:**

- get()** - **Get one or more DOM elements by selector**

- contains(text)** - **Get the DOM element containing the text,**
e.g. `cy.contains('Welcome')`

- find()** - **Get the descendent DOM elements of a selector.**

- e.g. `cy.get('.article').find('button')` // Yield the 'button' within '.article'

- click()** – **click a DOM element,** e.g. `cy.get('button').click()`

- select()** - **Select an <option> within a <select>**

- e.g. `cy.get('#paymenttype').select('Visa')`

- See <https://docs.cypress.io/api/api/table-of-contents.html>

Cypress statements

cy.get(. . .selector . . .).should(...assertion/expectation)

- **Selector: Based on CSS/JQuery style.**
 - **Id**, e.g. `cy.get('#heading')`
 - **CSS Class**, e.g. `cy.get('.info-message')`
 - **Tag**, e.g. `cy.get('input')`
 - **Attributes**, `cy.get('button[type=submit]').click()`
 - **The data-test attribute.**
 - **nth-child**, e.g. **get the 8th column of the 3rd row in a table**
`cy.get('tbody').find('tr:nth-child(3)').find('td:nth-child(8)')`
 - **These can be combined**, e.g. `div.container` (the `div` tag with CSS class `.container`)

Cypress Test Runner

- **Main features:**
 - **Tests run** inside the browser.
 - **Full access to browser resources – DOM, cookies, local storage.**
 - **Framework-agnostic.**
 - **Flake-free test execution.**
 - **Auto retries commands (e.g. get()) to cope with slow DOM construction.**
 - **Deals with unpredictable nature of the web.**
 - **Supports time-travel for convenient debugging.**

\$ npx cypress open

(Interactive runner)

The screenshot displays the Cypress interactive runner interface. On the left, the 'INTEGRATION TESTS' section lists 'contact-add.spec.js' and 'contact-delete.spec.js'. A red arrow points to 'contact-add.spec.js'. The main area shows the test suite 'Contact Add' with a status of '1 passed, 0 failed, 0 pending'. Below this, the 'TEST' steps are listed:

- 1 GET span.badge
- 2 - CONTAINS 5
- 3 GET input[placeholder=Name]
- 4 - TYPE user X
- 5 GET input[placeholder=Address]
- 6 - TYPE 22 main street
- 7 GET input[placeholder="Phone No."]
- 8 - TYPE 055 123456
- 9 GET button
- 10 - CONTAINS Add Contact
- 11 - CLICK
- 12 GET span.badge
- 13 - CONTAINS 6

On the right, the application view shows a contact list with five contacts:

Contact 1	Contact 2	Contact 3	Contact 4
123 Test St 132-3212	23 Main St 934-4329	4 Lower St 432-5832	49 Upper Street 934-4290

Below these, there is a section for 'Contact 5' and 'user X' with their respective details and 'Edit'/'Delete' buttons.

Version 3.1.5 | Changelog

- **Time-travel – Step through test code to track UI state.**

The screenshot shows the Cypress test runner interface. On the left, the 'Tests' panel displays a test suite 'Contact Add' with a test 'allows a contact be added'. The test steps are listed, and the step '- TYPE 22 main street' is highlighted with an orange box. An orange arrow points from this step to the '22 main street' input field in the 'Add Contact' form on the right. The form also includes a 'Phone No.' field. Below the form, there are five contact cards: Contact 1 (123 Test St, 132-3212), Contact 2 (23 Main St, 934-4329), Contact 3 (4 Lower St, 432-5832), Contact 4 (49 Upper Street, 934-4290), and Contact 5 (4 High Street, 933-3390). Each card has 'Edit' and 'Delete' buttons. At the bottom right, a 'DOM Snapshot: after' button is visible.

Selectors

cy.get(...selector...).should(...assertion/expectation)

- **Has impact on test brittleness.**
 - **Small changes to CSS or HTML structure can cause test failure.**
- **Use data-test attribute, where possible, to avoid brittle tests, e.g.**

```
<input data-test="name" type="text" className="form-control" onChange={....} />
```

```
cy.get('input[data-test=name]').type("Joe Bloggs")
```

- Selector Playground – **good learning aid.**

contactList

localhost:8080/_/#/tests/integration/contact-add.spec.js

1 2.83

http://localhost:8080/ 1000 x 660

Contact Add

✓ allows a contact be added

cy.get(':nth-child(2) > .form-control')

Contact List 6

Add Contact

Name

Address

Phone No.

Contact 1	Contact 2	Contact 3	Contact 4
Contact 1	Contact 2	Contact 3	Contact 4
123 Test St	23 Main St	4 Lower St	49 Upper Street
132-3212	934-4329	432-5832	934-4290

Cypress Test Runner

- **Headless mode:**
\$ npx cypress run
- **Runs all tests.**
- **Ideal for CI (Continuous Integration) environment.**
- **Generates video recordings (default; configurable).**
 - **.mp4 file type.**
 - **Facilitates sharing and project visibility.**
 - **Dashboard service (Publish recordings)**

Commands are in a queue

```
it('adds 2 todos', () => {  
  cy.visit('http://localhost:3000')  
  cy.get('.new-todo')  
    .type('learn testing{enter}')  
    .type('complete lab{enter}')  
  cy.get('.todo-list li')  
    .should('have.length', 2)  
})
```

